

A BEHAVIOUR OF RSA ALGORITHM USING 19 BIT PRIME DIGIT

GURPREETKAUR & VISHAL ARORA

Department of Computer Science & Engineering SBSSTC, Ferozepur, Punjab, India

ABSTRACT

RSA cryptographic algorithm used to encrypt and decrypt the messages for the citation of message in user's Communication over the secure transmission channel like internet. it will also throw some light on the flaws of some other public key cryptographic algorithms in comparison to RSA algorithm. RSA algorithm will be used to generate key pairs (private key and public key), which are generated using 19 digit prime value, which will be used to encrypt and decrypt the messages. Algorithm is implemented using Big Integer which makes Algorithm work efficiently and time complexity is also reduced

KEYWORDS: Cryptography, RSA, Big Integer, Secure Communication Channel

INTRODUCTION TO CRYPTOGRAPHY

In information technology, the possibility that the information stored in personal computer or the information being transferred through network of computer or internet being read by other people is very high. This causes a major concern for privacy, identity theft, security, military communication etc. we need a system which is very secure and efficient, in which data is read by only those persons who are authorized to it. A technique called cryptography is implemented which provides secure and efficient communication between client and server. The main idea behind the cryptography is, the original message is converted into unreadable form or that is readable but makes no sense of what the original message is. At server side the unreadable form is then converted into original message. All is done using two keys known as public key and private key.

RSA Algorithm

The RSA[1] public key cryptosystem was invented by R. Rivest, A. Shamir and L. Adleman[2]. Cryptography is the science of information security. Cryptography includes techniques such as microdots, merging words with images, and other ways to hide information in storage or transit. However, in today's computer-centric world, cryptography is most often associated with scrambling plain text (ordinary text, sometimes referred to as clear text) into cipher text (a process called encryption), then back again (known as decryption). In cryptography, encryption is the process of encoding messages (or information) in such a way that eavesdroppers or hackers cannot read it, but that authorized parties can. In an encryption scheme, the message or information (referred to as plaintext) is encrypted using an encryption algorithm, turning it into an unreadable cipher text. This is usually done with the use of an encryption key, which specifies how the message is to be encoded. Any adversary that can see the cipher text, should not be able to determine anything about the original message. An authorized party, however, is able to decode the cipher text using a decryption algorithm, that usually requires a secret decryption key, that adversaries do not have access to. For technical reasons, an encryption scheme usually needs a key-generation algorithm, to randomly produce keys.

A. PUBLIC KEY CRYPTOSYSTEM

A user wishing to exchange encrypted messages using a public-key cryptosystem would place their public encryption procedure, E , in a public file. The user's corresponding decryption procedure, D , is kept confidential. Rivest, Shamir, and Adleman provide four properties that the encryption and decryption procedures have[3]:

- Deciphering the enciphered form of a message M yields M . That is, $D(E(M)) = M$
- E and D are easy to compute.
- Publicly revealing E does not reveal an easy way to compute D . As such, only the user can decrypt messages which were encrypted with E . Likewise, only the user can compute D efficiently.
- Deciphering a message M and then enciphering it results in M . That is, $E(D(M)) = M$

As Rivest, Shamir, and Adleman point out, if a procedure satisfying property (3) is used, it is extremely impractical for another user to try to decipher the message by trying all possible messages until they find one such that $E(M) = C$. A function satisfying properties (1) - (3) is called a "trap-door one-way function". It is called "one-way" because it is easy to compute in one direction but not the other. It is called "trap-door" because the inverse functions are easy to compute once certain private, "trap-door" information is known

B. KEY GENERATION

We have generated encryption key using the following technique: We started off by generating two large prime numbers ' p ' and ' q ', [3] of about the same size in bits. The prime numbers should be of very large digit that can be taken. Next, compute ' n ' where $n = p * q$, and x such that, $x = (p - 1) * (q - 1)$. We select a small odd integer less than x randomly, which is relatively prime to it. Finally we find out the unique multiplicative inverse of e modulo x , and name it ' d '. In other words, $e * d = 1 \pmod{x}$, and of course, $1 < d < x$. Now, the public key is the pair (e, n) and the private key is d .

C. RSA ENCRYPTION

Suppose Bob wishes to send a message (say ' m ') to Alice. To encrypt the message using the RSA encryption scheme, Bob must obtain Alice's public key pair (e, n) . The message to send must now be encrypted using this pair (e, n) . However, the message ' m ' must be represented as an integer in the interval $[0, n-1]$. To encrypt it, Bob simply computes the number ' c ' where $c = m^e \pmod{n}$. Bob sends the cipher text c to Alice.

D. RSA DECRYPTION

To decrypt the cipher text c , Alice needs to use her own private key d (the decryption exponent) and the modulus n . Simply computing the value of $c = m^e \pmod{n}$ yields back the decrypted message (m). Any article treating the RSA algorithm in considerable depth proves the correctness of the decryption algorithm. And such texts also offer considerable insights into the various security issues related to the scheme. Our primary focus is on a simple yet flexible implementation of the RSA cryptosystem that may be of practical value.

INTEGER FACTORIZATION AND RSA PROBLEM

The security of the RSA cryptosystem is based on two mathematical problems: the problem of factoring large numbers and the RSA problem. Full decryption of an RSA cipher text is thought to be infeasible on the assumption that both of these problems are hard, i.e., no efficient algorithm exists for solving them. Providing security against partial

decryption may require the addition of a secure padding scheme. The RSA problem is defined as the task of taking the roots modulo a composite recovering a value m such that $c \equiv m^e \pmod{n}$, where (n, e) is an RSA public key and c is an RSA cipher text. Currently the most promising approach to solving the RSA problem is to factor the modulus n . With the ability to recover prime factors, an attacker can compute the secret exponent d from a public key (n, e) , then decrypt c using the standard procedure.

To accomplish this, an attacker factors n into p and q , and computes $(p-1)*(q-1)$ which allows the determination of d from e . No polynomial-time method for factoring large integers on a classical computer has yet been found, but it has not been proven that none exists. Integer factorization is a point of discussion here in research intended to present this problem.

Rivest, Shamir and Adleman note that Miller has shown that assuming the Extended Riemann Hypothesis (though others call it the Generalized Riemann Hypothesis) - finding d from n and e is as hard as factoring n into p and q (up to a polynomial time difference). However, Rivest, Shamir and Adleman note (in section IX / D of their paper) that they have not found a proof that inverting RSA is equally hard as factoring.

How secure is a communication using RSA?

Cryptographic methods cannot be proven secure. Instead, the only test is to see if someone can figure out how to decipher a message without having direct knowledge of the decryption key. The RSA method's security rests on the fact that it is extremely difficult to factor very large numbers. If 100 digit numbers are used for p and q , the resulting n will be approximately 200 digits. The fastest known factoring algorithm would take far too long for an attacker to ever break the code. Other methods for determining d without factoring n are equally as difficult. Any cryptographic technique which can resist a concerted attack is regarded as secure. At this point in time, the RSA algorithm is considered secure.

SECURITY OF RSA

Three possible approaches to attacking the RSA algorithm are as follows: Brute Force. This involves trying out all the possible private keys. Mathematical attacks: There are several approaches, all equivalent in effect to integer factoring the product of two prime numbers.

TIMING ATTACKS

These depend on the running time of the decryption algorithm. Choosing large p and q values can prevent such attacks. Security of RSA thus lies in choosing the value n , which makes such attacks extremely difficult. As of 2010, the largest (known) number factored by a general-purpose factoring algorithm was 768 bits long (see RSA-768), using a state-of-the-art distributed implementation. RSA keys are typically 1024–2048 bits long. Some experts believe that 1024-bit keys may become breakable in the near future (though this is disputed); few see any way that 4096-bit keys could be broken in the foreseeable future. Therefore, it is generally presumed that RSA is secure if n is sufficiently large. If n is 300 bits or shorter, it can be factored in a few hours on a personal computer, using software already freely available. Keys of 512 bits have been shown to be practically breakable in 1999 when RSA was factored by using several hundred computers and are now factored in a few weeks using common hardware.^[11] Exploits using 512-bit code-signing certificates that may have been factored were reported in 2011.^[12] Keys of 1024 bits have been implemented in July 2012, which is a very large bit prime digit which makes code very lengthy and time consuming. It also does not provide flexibility to the system. It can be only implemented with a fixed key size [6].

PRESENT WORK

Present work of RSA algorithm include the 19 bit prime digit. Which are implemented using big integer. The implementation of RSA is done in java. key pairs are generated in java as follows :Public key consists of modulus and a public exponent Private key consists of same modulud plus private exponent The idea behind is as follows We create an instance of generate Prime Numbers() for generating prime number. We call generate Public Private Keys() on the latter to pull out the public and private keys Implementation of RSA: the main point in implementation is how key pairs are generated in RSA. Generate a public key (e) and private key(d) by choosing two large prime number p and q each around 19 bits. Multiply p and q i.e $n=p*q$. p and q are secret and n is public Generate a public key taking a number e, i.e $\phi(n)=(p-1)(q-1)$.Generate a private key taking a number d, which is a multiplicative inverse of e mod $\phi(n)$. Encryption: $c=m^e \bmod n$.

- Decryption: $c^d \bmod n$.
- Key generation: For Example,

Let us assume that the two prime numbers be 3 and 5. So, this implies $n=3*5=15$. And, $x=(3-1)*(5-1)=2*4=8$.Let us suppose that the randomly generated number be 7 which is between 1 and x we can see.Now, The H.C.F. of the number and x will be H.C.F.(8,7) is 1(one).And the last step is to find the modulus of randomly generated number and x, $\text{mod}(8,7)$ is 1 i.e. remainder is 1 .So at the end we concluded that the public key is (7,15) and private key is (1,15).Decryption is the technique of converting the cipher text into plain text which is the original text. Decryption requires a secret key or password. So we can do this by using the private key. Simply computing the value of $c=m^d \bmod n$ yields back the decrypted message (m).The main thing which we had used in our project is the use of Big Integer [4], which helps us to take a prime number up to 19-digit and this is not very easy to break the code for 19-digit prime number so we can say that it is more secure and powerful.

RESULTS AND DISCUSSIONS

In given Table 1, Observation of cryptographic algorithm is shown. From this we get Computationally Equivalent Keys description of RSA as compare to other public key cryptographic algorithms which had implemented on 19 bit big prime number.

Table 1: Observation Description of RSA

| 19 Digit Prime | Keys |
|----------------|--------------------------------|
| E2 | Public [A07F] private[A07F] |
| E3 | Public [3F95] private[988D] |
| B4 | Public [F01F] private[A03F] |
| B5 | Public [C02F] private[A04F] |

Describes the nature of public key cryptographic algorithm Where the value of keys and their strength depend upon Standard on which they are implemented, experimentation report observed on variation of bits of prime number to be used for the fast implementation and better results.

CONCLUSIONS

Implementation of RSA Algorithm on FPGA use 1024 bit key size according to Spec standard of ANSI. 20 digit prime which is very large in size, as a result process is time consuming. Key strength reduces for such a large prime, so after the experimentation observation without effecting key strength public and private key are produced for 19 digit prime, which it's a big prime number, may maintain the strength results are observed over large scale using Eclipse Indigo for java implementation of such a huge cryptosystem. So, Big Integer which makes Algorithm works as efficiently and less time complex or we can say gives satisfactory results.

REFERENCES

1. Bahadori, M.; Mali, M.R.; Sarbishei, O. Atarodi M. Sharifkhani, M."symmetric approach to public key cryptographic algorithm"
2. RajorshiBiswas, ShibdasBandyopadhyay, Anirban Banerjee "A analysis approach to prime field taxonomy using MATLAB"
3. A fast implementation of the RSA algorithm using the gmp library, RajorshiBiswas, ShibdasBandyopadhyay, AnirbanBanerjee IIIT-Calcutta
4. B. Schneier. Applied Cryptography. John Willey and Sons,1996
5. William Stallings, Cryptography and Network Security-Principles and Practice second edition, Prentice Publications.
6. Singhal, Mukesh and Shivaratri, Niranjana G Advanced Concepts in Operating Systems, McGraw- Hill.
7. Primes is in P", M. Aggarwal, et. al. www.cse.iitk.ac.in/news/primalty.pdf
8. Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public -Key cryptosystems", Communications of the ACM, Vol. 21, No.2
9. Bidzos, Jim, "Threats to Privacy and Public Keys for Protection", COMPCON IEEE . Computer Society press.

